Lab Report 1

Zhian Lin, Sonnelly Cheong

```r
## Read data here
spotify <- read.csv("spotify_songs.csv", header=TRUE)

## call required libraries
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse

## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------------------------------------------ tidyverse_confli
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library('ggridges')

## Set theme to something else (optional)
theme_set(theme_minimal())
```

Describe dataset:

Dataset description: A large dataset from Spotify. Containing information about 32000 songs; album, artist, genre and subgenre together with extracted properties of songs using an ML algorithm (danceability, tempo, liveness, instrumentalness, key, etc.).

```r
str(spotify)
```

```
## 'data.frame':    32833 obs. of  23 variables:
##  $ track_id                : chr  "6f807x0ima9a1j3VPbc7VN" "0r7CVbZTWZgbTCYdfa2P31" "1z1Hg7Vb0A
##  $ track_name              : chr  "I Don't Care (with Justin Bieber) - Loud Luxury Remix" "Memo
##  $ track_artist            : chr  "Ed Sheeran" "Maroon 5" "Zara Larsson" "The Chainsmokers" ...
##  $ track_popularity        : int  66 67 70 60 69 67 62 69 68 67 ...
##  $ track_album_id          : chr  "2oCs0DGTsRO98Gh5ZSl2Cx" "63rPSO264uRjW1X5E6cWv6" "1HoSmj2eLc
##  $ track_album_name        : chr  "I Don't Care (with Justin Bieber) [Loud Luxury Remix]" "Memo
##  $ track_album_release_date: chr  "2019-06-14" "2019-12-13" "2019-07-05" "2019-07-19" ...
##  $ playlist_name           : chr  "Pop Remix" "Pop Remix" "Pop Remix" "Pop Remix" ...
##  $ playlist_id             : chr  "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1D
##  $ playlist_genre          : chr  "pop" "pop" "pop" "pop" ...
##  $ playlist_subgenre       : chr  "dance pop" "dance pop" "dance pop" "dance pop" ...
## $ danceability            : num  0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542 0.594 0.642 ..
## $ energy                  : num  0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903 0.935 0.818 ..
## $ key                     : int  6 11 1 7 1 8 5 4 8 2 ...
## $ loudness                : num  -2.63 -4.97 -3.43 -3.78 -4.67 ...
## $ mode                    : int  1 1 0 1 1 1 0 0 1 1 ...
## $ speechiness             : num  0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623 0.0434 0.0565
## $ acousticness            : num  0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187 0.0335 0.0249
## $ instrumentalness        : num  0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00 4.83e-06 3.97e
## $ liveness                : num  0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111 0.637 0.0919
## $ valence                 : num  0.518 0.693 0.613 0.277 0.725 0.585 0.152 0.367 0.366 0.59 ..
## $ tempo                   : num  122 100 124 122 124 ...
## $ duration_ms             : int  194754 162600 176616 169093 189052 163049 187675 207619 19318
```

The dataset contains information for 32833 songs (subject) and the data is organized in 23 columns (variables). By using the head function I am able to see that exact column names and how the data is being organized. From this I can tell that the whether the variable is categorical or numerical. Initially, out of 23 variables, 10 of them are categorical variables and 13 of them are numerical

Categorical variables contain:

track_id, track_name, track_artist, track_album_id, track_album_name, track_album_release_date, playlist_name, playlist_id, playlist_genre, playlist_subgenre.

Numerical variables contain:

track_popularity, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms

However the two variables key and mode that are initially being identified as numerical variables should be categorical. Since key identifies and describes the pitch of a song and modes are a type of scale with distinct melodic characteristics. Therefore, convert to factor.

```r
spotify$key <- as.factor(spotify$key)
spotify$mode <- as.factor(spotify$mode)
```

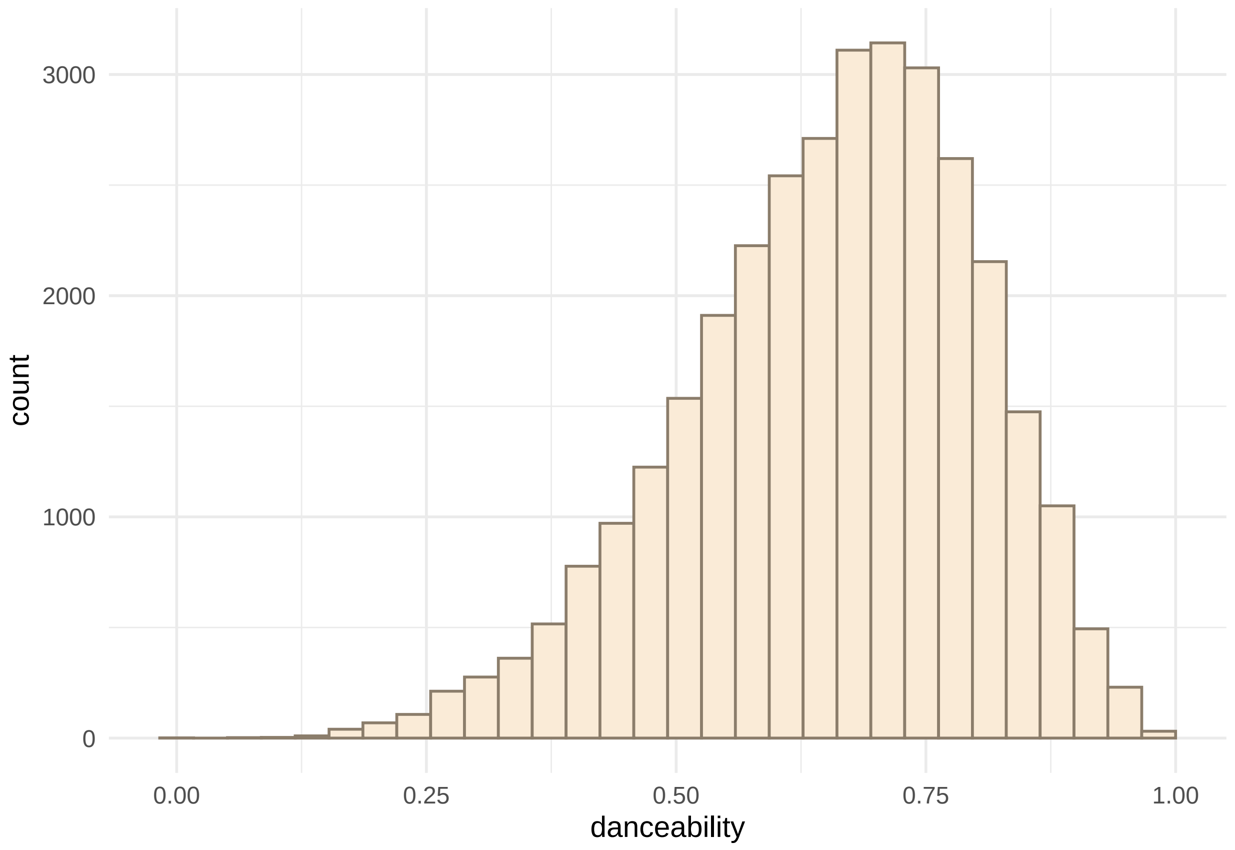
after this convertion there will be 12 categorical and 11 numerical variables.

1. Histogram & Density

Variable selected: danceability (numerical), playlist_genre (categorical)

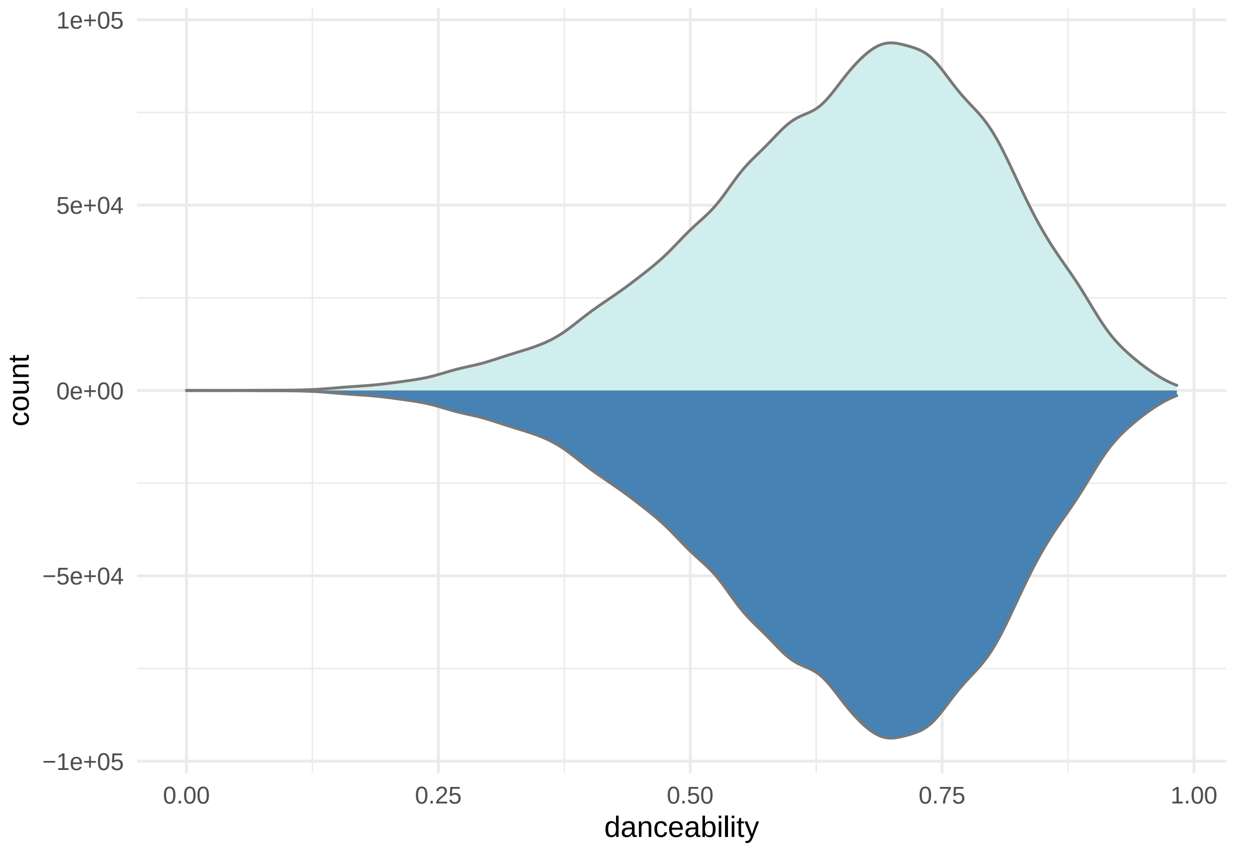
```r
ggplot(data = spotify) +
  geom_histogram(aes(x = danceability), fill = 'antiquewhite', colour = 'bisque4')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



from this histogram we were able to observe that the danceability of the 32080 songs are concentrated at the danceability of 0.50 - 0.875. There are around 3000 songs with the danceability of around 0.75. From this distribution of data I can conclude that a lot of track is suitable fr dancing based on a combination of musical elements including tempo, rythm stability, beat strength, and overall regularity.

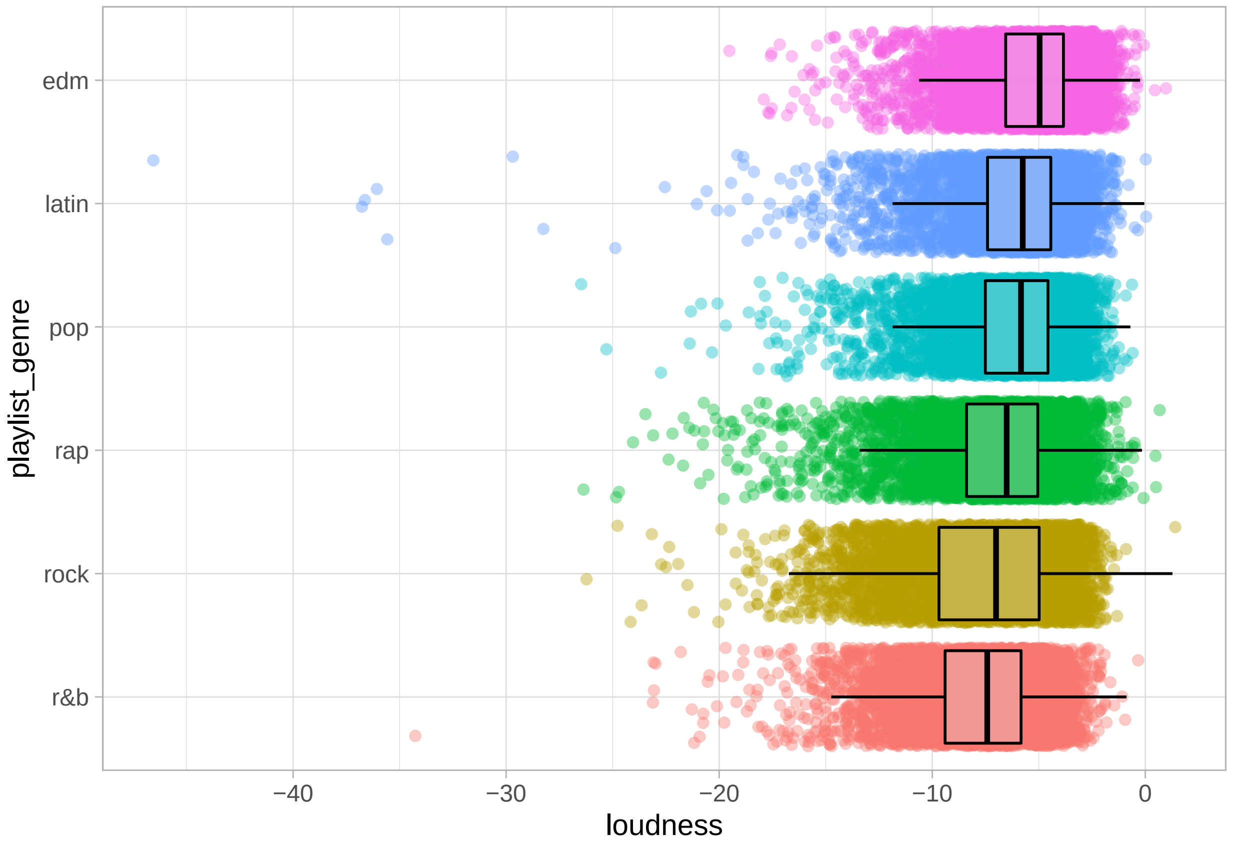
```
ggplot(spotify) +
  geom_density(data= subset(spotify, playlist_genre = 'pop'), aes(x = danceability,
    y = ..count..), fill="lightcyan2", colour='gray47') +
  geom_density(data= subset(spotify, playlist_genre = 'r&b'), aes(x = danceability,
    y = - ..count..), fill="steelblue", colour='gray47')
```



From this back to back density graph I can observe that for both playlist genres of r&b and pop the danceability are both concentrated at around 0.75 meaning that both playlist genres are danceable.

2. Boxplot

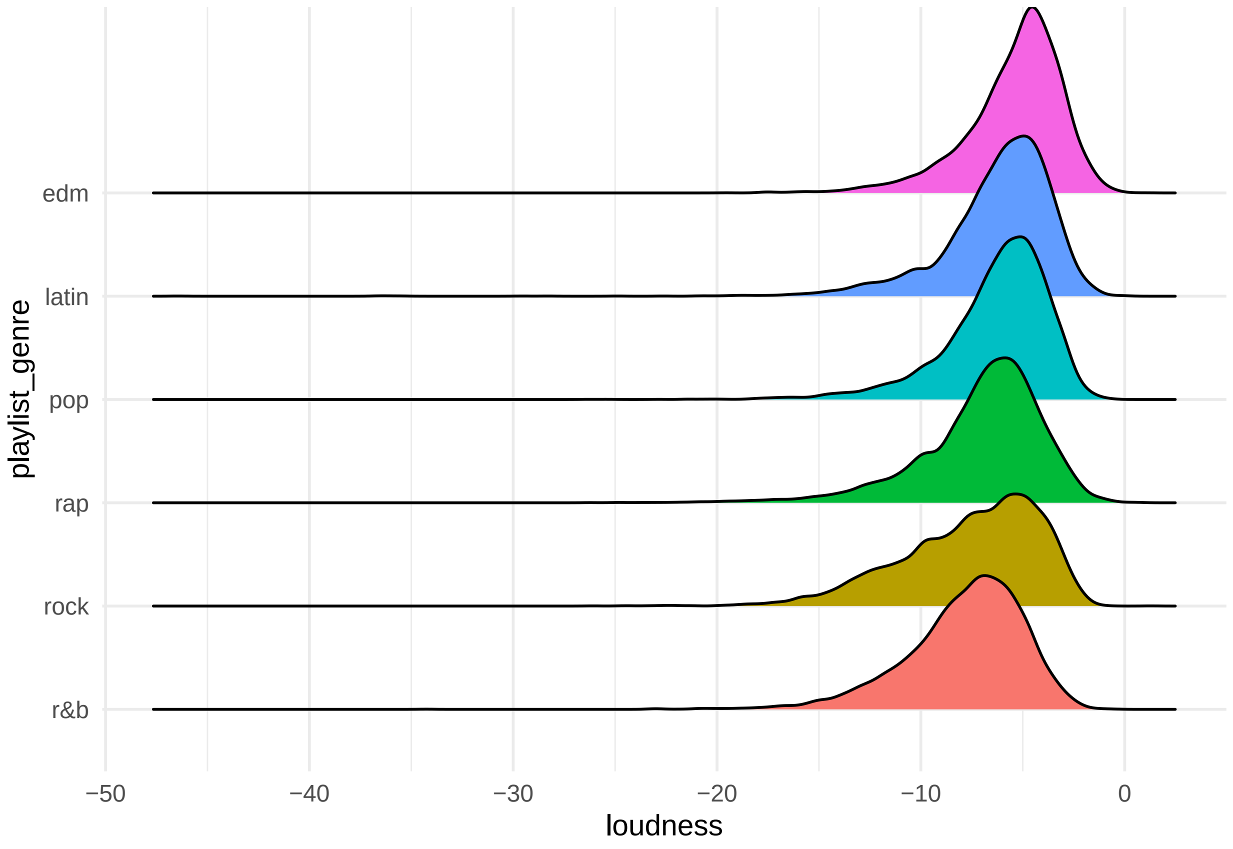
```
# Selecting loudness as the numeric value
# Selecting playlist_genre as the categorical value

playlist_genres <- unique(spotify$playlist_genre)
medians  <- sapply(playlist_genres, function(z) median(spotify$loudness
    [spotify$playlist_genre == z]))
    ord <- order(medians)  # the order

spotify$playlist_genre <- factor(spotify$playlist_genre , levels=playlist_genres[ord])

ggplot(spotify, aes(y=playlist_genre, x=loudness, color=playlist_genre)) +
  geom_jitter(alpha=.4, position = position_jitter(.2)) +
  geom_boxplot(color='black',fill=adjustcolor('grey90',.3),outlier.alpha = 0) +
  theme_light() +
  theme(legend.position = 'none')
```



Boxplot & Jitter Plot We can see the distribution of loudness for each playlist genre The box and whiskers give us information about the different quartiles, while the jitter plot gives us a sense of how many playlists have a certain loudness with a given genre. As displayed on the plots, there are some outliers. There are particularly more outliers for the latin playlist genre. The median for loudness of each playlist genre appears to fall somewhere in between -10 and -5db. The edm genre has the highest median followed by latin, pop, rap, rock, and r&b. The rock playlist genre has the largest interquartile range.

```
ggplot(spotify, aes(y=playlist_genre, x=loudness, fill=playlist_genre)) +
  geom_density_ridges() +
  theme_minimal() +
  theme(legend.position = 'none')
```

```
## Picking joint bandwidth of 0.402
```



```
  ggtitle('Density Ridges Plot for Loudness')
```

```
## $title
## [1] "Density Ridges Plot for Loudness"
##
## attr(,"class")
## [1] "labels"
```

Density Plot We can see the continuous distribution of loudness for each playlist genre. Although the typical range for loudness is between -60 and 0db, we can see that majority fo the loudness values are greater than -20db. We can observe that the peak of each distribution is typically in between -10 and 0. The concentration of values for all genres appear to lie between -10 and 0db. The peak of the distribution for the r&b genre appears to be at the lowest loudness. The steepness of each curve can also be observed. For the rock genre, it has a 'wider' distribution and so there is a larger proportion of rock songs that cover a wider range.

3. Aggregating Data

Categorical variable selected: variable 1: track_artist variable 2: playlist_genre numerical value: track_release_date

```
# select
select(spotify, track_artist) %>% group_by(track_artist) %>% summarize(Songs = n()) %>% head()
```

part 1

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 6 x 2
##   track_artist Songs
##   <chr>        <int>
## 1 _tag             1
## 2 -M-              1
## 3 !!!              2
## 4 !deladap         5
## 5 .Sinh            3
## 6 .SØN             1
```

```
select(spotify, playlist_genre) %>% group_by(playlist_genre) %>% summarize(Songs = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 6 x 2
##   playlist_genre Songs
##   <fct>          <int>
## 1 r&b             5431
## 2 rock            4951
## 3 rap             5746
## 4 pop             5507
## 5 latin           5155
## 6 edm             6043
```

After grouping the artist together and applied the summarize function we were able to see the amount of songs that a particular artist has had in this list. After applying the group by function to the playlist_genre, we were able to determine the total amount of songs distributed across each playlist.

```
spotify %>%
  rename(artist=track_artist, release_date= track_album_release_date) %>%
  group_by(release_date, artist) %>%
  summarize(track= n()) %>%
  arrange(release_date)
```

```
## `summarise()` regrouping output by 'release_date' (override with `.groups` argument)
```

```
## # A tibble: 21,732 x 3
## # Groups:   release_date [4,530]
##    release_date artist                 track
##    <chr>        <chr>                  <int>
##  1 1957-01-01   Ray Charles                1
##  2 1957-03      Little Richard             1
##  3 1958-03-21   Elvis Presley              1
##  4 1960         Ella Fitzgerald            1
##  5 1960         Etta James                 2
##  6 1960         Sam Cooke                  1
##  7 1961-10-26   Chavela Vargas             1
##  8 1962         Booker T. & the M.G.'s     2
##  9 1963         Darlene Love               1
## 10 1963-03-22   The Beatles                1
## # ... with 21,722 more rows
```

By applying the groupby, rename, summarize and arrange, I was able to see the release date of each song track in acsending order with with the artist name and the number of song track that they have released on that day which made into this list.

4. Visualizing Categorical Variables

```
music_genre <- spotify %>%
  rename(genre= playlist_genre, subgenre= playlist_subgenre) %>%
  group_by(genre,subgenre) %>%
  summarize(songs= n()) %>%
  arrange(genre)
```

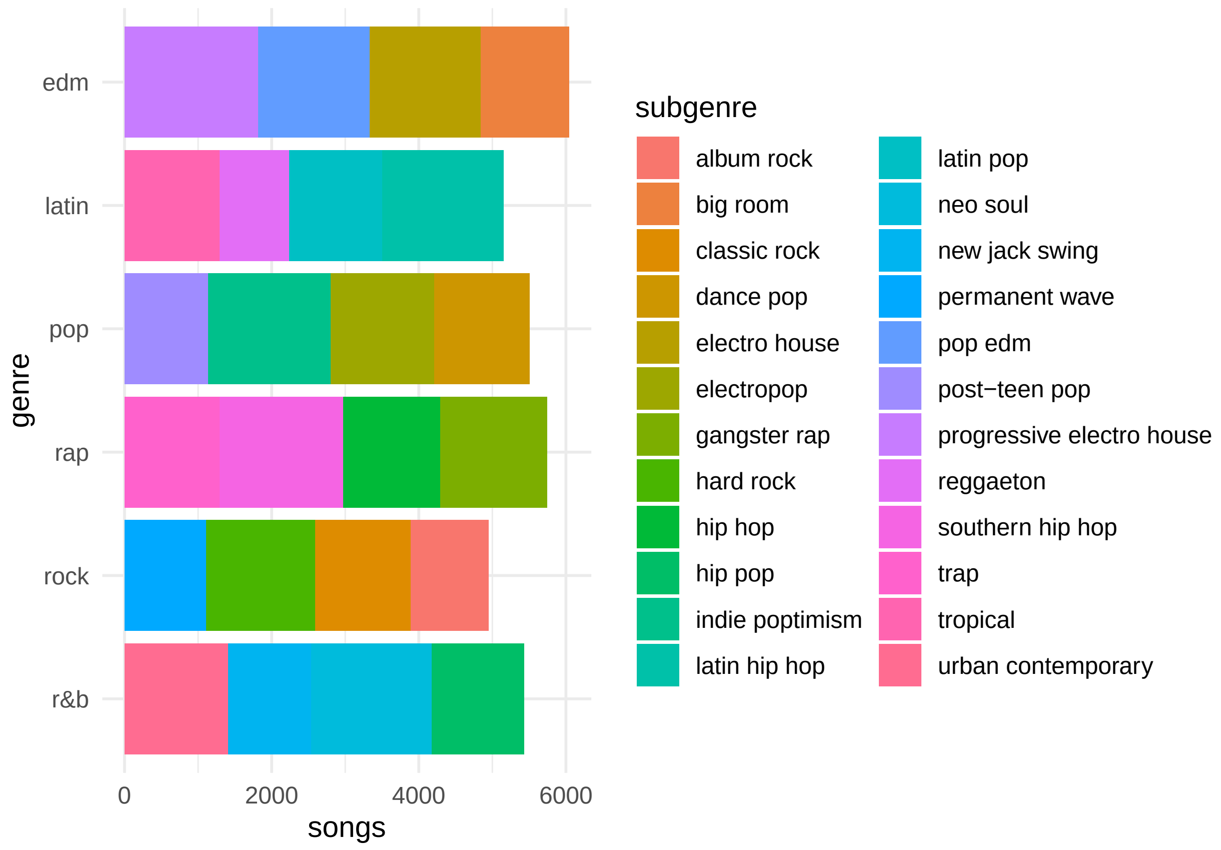
```
## `summarise()` regrouping output by 'genre' (override with `.groups` argument)
```

```
music_genre
```

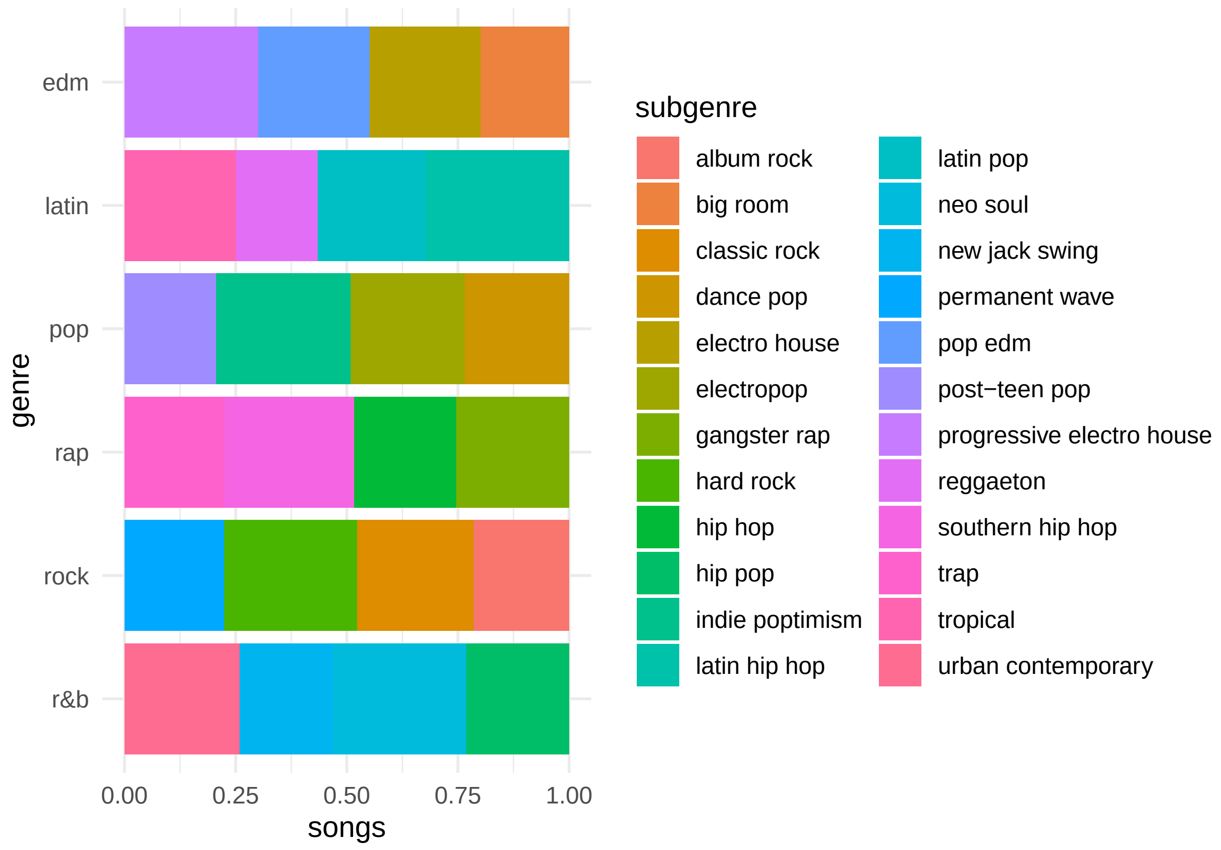
```
## # A tibble: 24 x 3
## # Groups:   genre [6]
##    genre subgenre           songs
##    <fct> <chr>              <int>
##  1 r&b   hip pop             1256
##  2 r&b   neo soul            1637
##  3 r&b   new jack swing      1133
##  4 r&b   urban contemporary  1405
##  5 rock  album rock          1065
##  6 rock  classic rock        1296
##  7 rock  hard rock           1485
##  8 rock  permanent wave      1105
##  9 rap   gangster rap        1458
## 10 rap   hip hop             1322
## # ... with 14 more rows
```

This gives a detailed overview before the visualization of how each subgeneres is being distributed across individual genres. For example the genre lating has four subgenres under it they are latin hip hop with 1656 songs, latin pop with 1262 songs, reggaeton with 949 songs and tropical with 1288 songs.

```
ggplot(music_genre, aes(y = genre, x = songs, fill = subgenre)) +
  geom_bar(stat = 'identity')
```



```r
ggplot(music_genre, aes(y = genre, x = songs, fill=subgenre)) +
  geom_bar(stat = 'identity', position='fill')
```

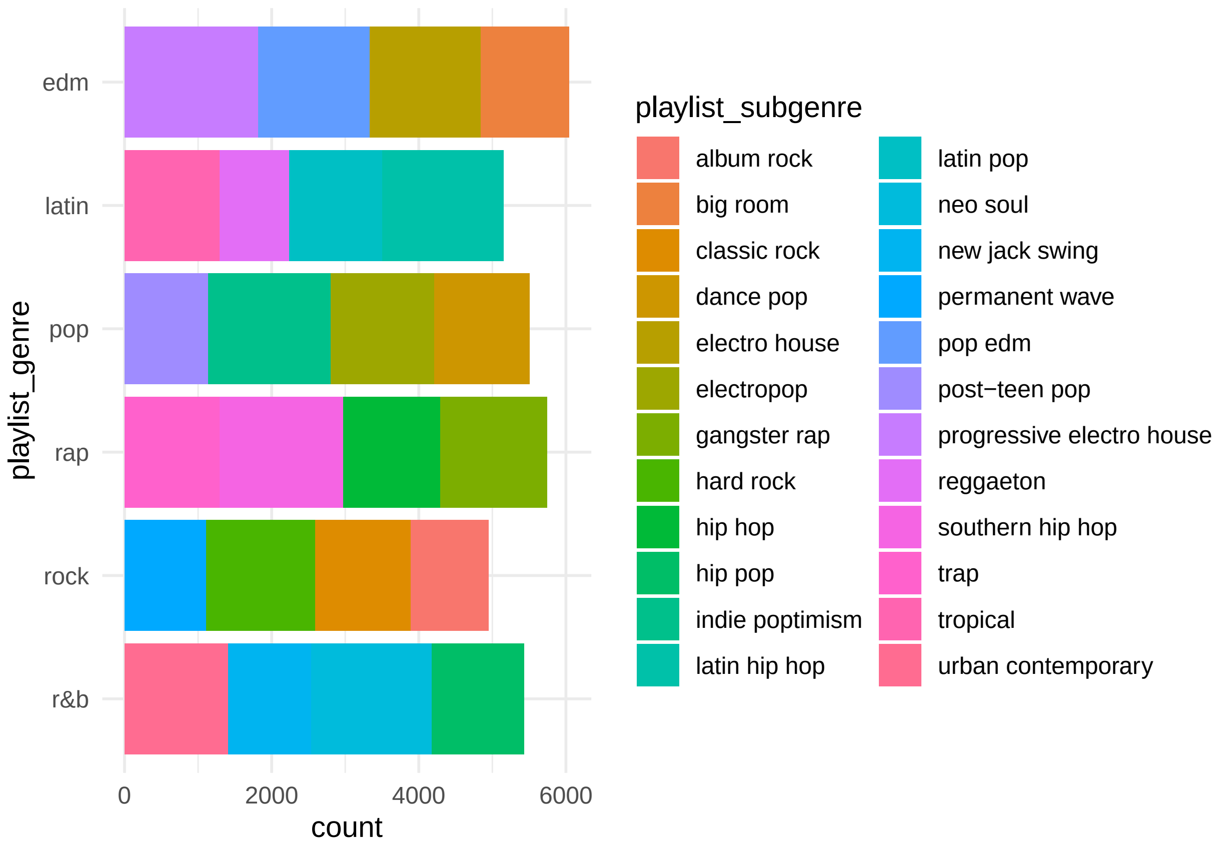


>

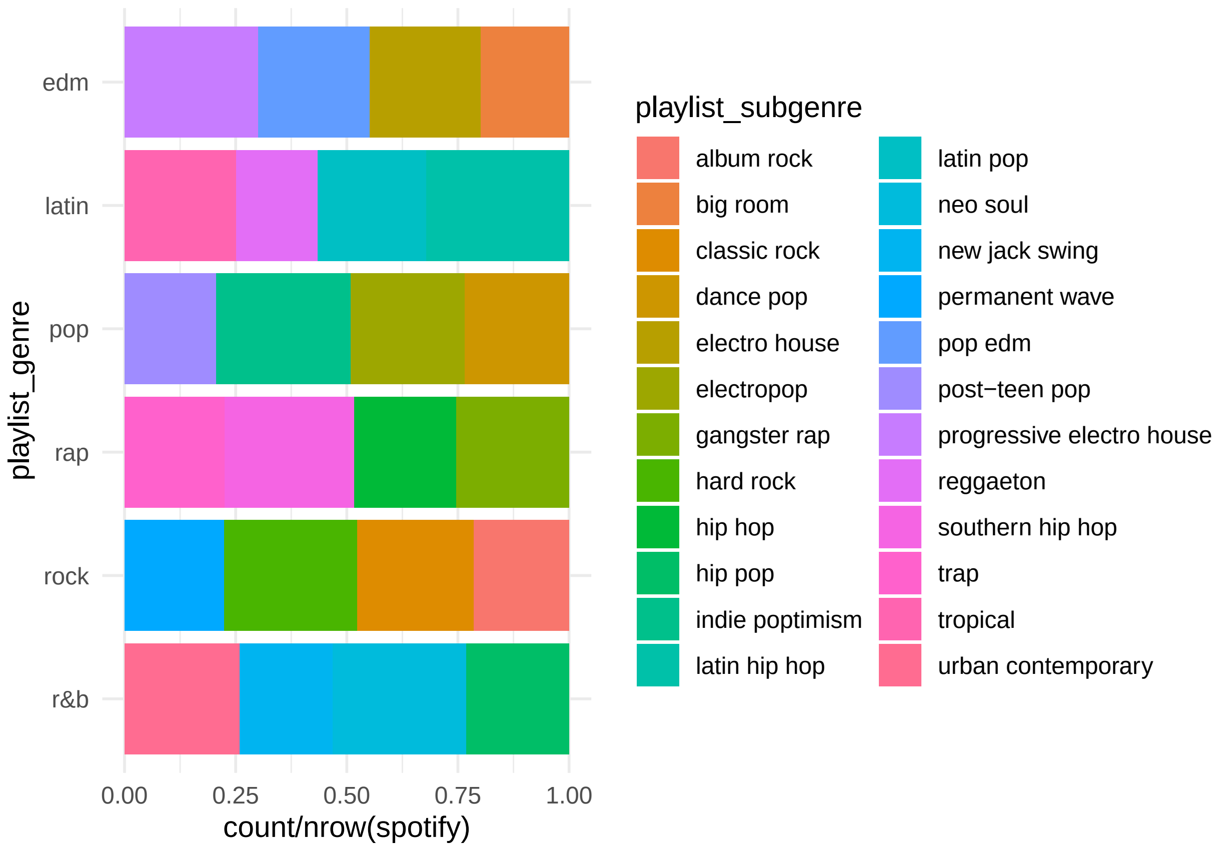
From the above two graphs, we were able to observe the distributions of the subgeneres in the generes. For example for the rock genere is being spanned by four subgenres that are permanent wave, hip hop, classic rock and album rock. We can also tell the porportions that those subgenres are distributed within each genre. For example, in the genre edm, the four subgenres spreads out quit evenly with progressive elector house haveing a little bit more weight than the other subgenres.

Ploting using stat(count)

```r
ggplot(spotify, aes(y = playlist_genre, x = stat(count), fill = playlist_subgenre)) +
  geom_bar()
```



```
ggplot(spotify, aes(y = playlist_genre, x = stat(count)/nrow(spotify),
  fill=playlist_subgenre)) +
  geom_bar(position = 'fill')
```

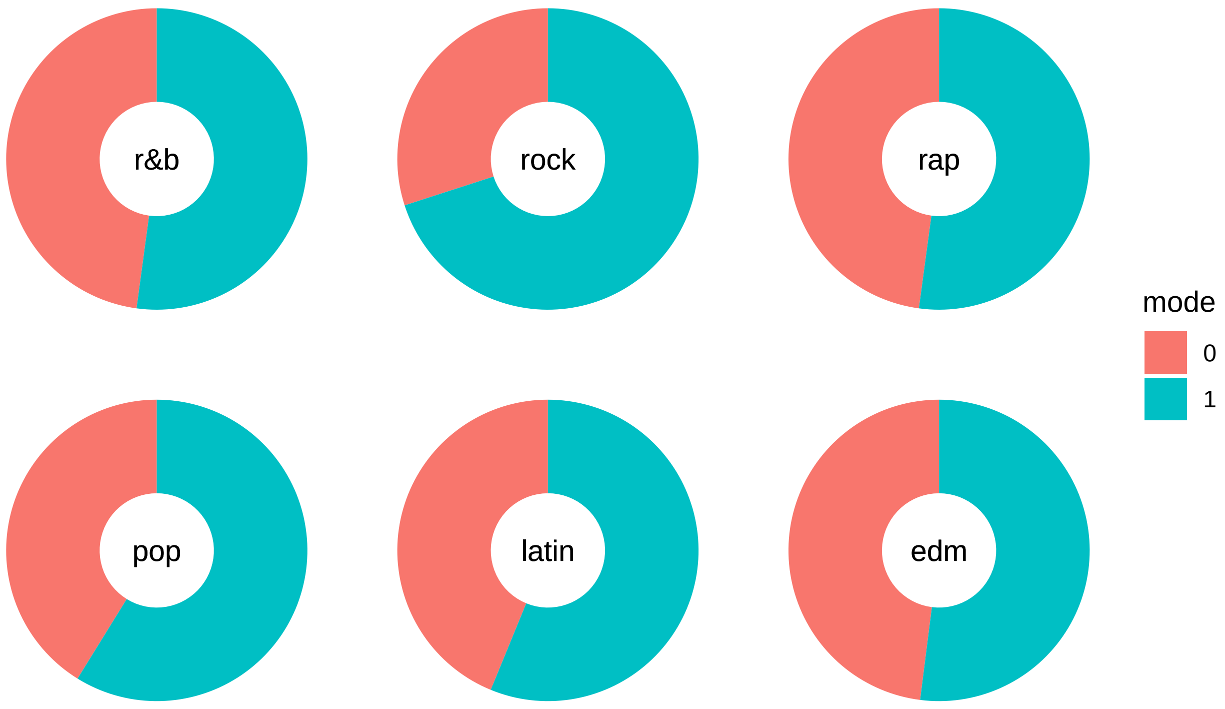

edm
latin
pop
rap
rock
r&b
playlist_genre
0.00
0.25
0.50
0.75
1.00
count/nrow(spotify)
playlist_subgenre
album rock
big room
classic rock
dance pop
electro house
electropop
gangster rap
hard rock
hip hop
hip pop
indie poptimism
latin hip hop
latin pop
neo soul
new jack swing
permanent wave
pop edm
post−teen pop
progressive electro house
reggaeton
southern hip hop
trap
tropical
urban contemporary

5. Other Visuals

```
aggDat <- group_by(spotify, playlist_genre, mode) %>% summarise(track_id = n())
```

```
## `summarise()` regrouping output by 'playlist_genre' (override with `.groups` argument)
```

```
ggplot(aggDat) +
      geom_col(aes(x = 1, y = track_id, fill = mode), position = "fill") +
      facet_wrap( ~ playlist_genre) +
      geom_text(aes(x = 0, y = 0, label = playlist_genre)) +
      coord_polar(theta = "y") +
      theme_void() +
      theme(strip.background=element_blank(),
          strip.text=element_blank())
```


r&b
rock
rap
pop
latin
edm
mode